REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1996	Interim, June 1995 - June 1996

4. TITLE AND SUBTITLE

Automated Evaluation of Tactical Network Protocols

5. FUNDING NUMBERS

P622783.094
JONO: 6TE710
CC: BFT00

6. AUTHOR(S)

Maria C. Lopez, Ann E. M. Brodeen, George W. Hartwig, Jr., and Michael J. Markowski

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
ATTN: AMSRL-IS-TP
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Decentralized battlefield command and control requires reliable and timely distribution of information. At present, distribution of digital information is limited by the low-bandwidth noisy channels inherent to combat net radios and heavy traffic demands, forcing commanders to make decisions from less than timely information. In the ideal communications network, each node would be smart enough to monitor network performance and, when necessary, adapt itself to make better use of the available bandwidth. The adaptive network node would employ a decision algorithm to modify configuration, routing, and protocol parameters based on measured network performance statistics and system requirements. Our research addresses the effects of noise and interference on communications channels and construction of network protocols that will be effective on the modern battlefield. The approach emphasizes use of actual hardware and controlled experimentation to explore alternative protocols. This report describes a suite of software to automatically execute a test design, and collect and apply preliminary data reduction procedures to baseline performance data for a prototype communications network.

14. SUBJECT TERMS

MORS, factorial design, tactical data buffers, communications channels, automated evaluation tactical network protocols

15. NUMBER OF PAGES

7

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

AUTOMATED EVALUATION OF TACTICAL NETWORK PROTOCOLS

Maria C. Lopez, Ann E. M. Brodeen,
George W. Hartwig, Jr., Michael J. Markowski

U.S. Army Research Laboratory
Aberdeen Proving Ground, Maryland 21005–5067

ABSTRACT

Decentralized battlefield command and control requires reliable and timely distribution of information. At present, distribution of digital information is limited by the low–bandwidth noisy channels inherent to combat net radios and heavy traffic demands, forcing commanders to make decisions from less than timely information. In the ideal communications network, each node would be smart enough to monitor network performance and, when necessary, adapt itself to make better use of the available bandwidth. The adaptive network node would employ a decision algorithm to modify configuration, routing and protocol parameters based on measured network performance statistics and system requirements. Our research addresses the effects of noise and interference on communications channels and construction of network protocols that will be effective on the modern battlefield. The approach emphasizes use of actual hardware and controlled experimentation to explore alternative protocols. This paper describes a suite of software to automatically execute a test design, and collect and apply preliminary data reduction procedures to baseline performance data for a prototype communications network.

BACKGROUND

The primary means of communications at low–echelon fighting units has been and continues to be voice data transmitted by combat net radios. Gradually, a requirement for digital data transmission is being inserted into the mission profile. Digital transmissions allow for compression and forward error correction and provide the ubiquitous computer with the information it requires. With this increasing requirement for digital transmissions, problems arise.

Modern combat net radios are typically line–of–sight FM low power instruments designed specifically for use at short range. Their bandwidth is very limited, typically 1200–2400 baud, although recent improvements in modem technology have pushed these numbers as high as 16 kilo–baud. These radios are commonly assembled into a single hop network of 6 to 12 users. Their effective use to date is testimony to the redundancy of the human language and the ability of the human brain to extract meaningful data from a noisy signal.

Our research addresses the effects of noise and interference on communications channels and construction of network protocols and procedures that will minimize delay and maximize throughput on the modern battlefield. The networks that are of particular interest to us have nodes with high computing power but weak, noisy, shared communications links. For this reason, our approach to communications emphasizes working intelligently at each node to limit or redirect the amount of information that must be passed along the communications channel. Each node is assumed to act independently to improve the effectiveness of the information exchange between nodes. Such a system of controls requires that each node be able to monitor the network traffic; decide whether performance is inadequate; and, if so, make an appropriate adjustment to the protocol.

A series of controlled experiments is being conducted to determine which communications protocol parameters and structural assumptions have the greatest impact on selected performance

measures as well as to explore the limitations of the software. To accomplish such objectives, it is required that a group of computers serving as battlefield nodes be synchronized, network parameters be initialized prior to each run, and collected data be made conveniently accessible to the user. As a result, software that performs the necessary tasks with minimal user intervention was developed.

TEST DESIGN AND APPROACH

Experimental design provides a means of deciding before any runs are made which particular configurations to examine so that the desired information can be collected with the least amount of testing. Carefully designed experiments are much more efficient than a "hit–or–miss" sequence of runs in which a number of alternative configurations are unsystematically tried just to see what happens.

When the number of factors is moderate, a factor–screening strategy, such as a factorial design, might be able to indicate which factors appear to be important, and more to the point, which factors are irrelevant and can be simply fixed at some reasonable level and omitted from further consideration. The software currently supports the fully automated execution of a modified 2^k factorial design. The 2^k factorial design is an economical strategy that requires choosing just two levels for each factor and then calls for runs at each 2^k possible factor–level of interest combination. The k factors selected for automation, retry time-out interval, window size, arrival rate, and message length are ones that can be easily modified. Past experimentation with actual hardware and a tactical communications protocol illustrated that network behavior is nonlinear in nature [Kaste, Brodeen, and Broome 1992]. A potential concern with the use of two–level factorial designs is the assumption of linearity in the factor effects. That is not to say that a 2^k system requires perfect linearity – this system works quite well even when the linearity assumption holds only very approximately. However, to provide protection against possible curvature in the response data, the 2^k factorial design has been modified by the addition of runs to be made at the center of the design. The replicate runs are added to the design center as the center points do not impact the usual effects estimates in a 2^k design. For more information refer to Montgomery [1991].

EXPERIMENTAL CONFIGURATION

There are three nodes, each of which is a SPARCbook 3 [SPARCbook 3 Series 1984]. Each contains a communications protocol and a scenario driver. The communications protocol includes data collection functions to log the sending and receipt of messages and acknowledgements as well as information on queues. The scenario driver provides the communications loading. The nodes are connected, via ethernet, to a SPARCstation 20 [SPARCstation 20 System 1994] that serves as the data storage and control node. The nodes are connected to Single Channel Ground and Airborne Radio System (SINCGARS) Combat Net Radios via Tactical Data Buffers (TDB). Resistor loads are used as the antennas to reduce the transmission range.

The TDB interfaces with data processing equipment using RS–232C. Two processing steps are performed to input data to the TDB: 1) any formatting bits, such as start, stop, and parity, are removed so that transmission time is not expended by unnecessary data; 2) the data are stored until the TDB can access the network.

When data are transmitted from the TDB, three levels of forward error correction are applied. First, Bose–Chandhuri–Hacquenghem (BCH) coding is applied at a 48/32 rate. Second, interleaved redundancy takes place with redundancies of 13 times, 5 times, or 1 time as determined by a front panel switch. Third, data randomization takes place. This assures an independence from the specific characteristics of any type radio when encryption is not being used.

The throughput rate is determined by the three forward error correction processes: interleaved redundancy of 1 time yields 10.66 kilobits per second (kbps), 5 times yields 2.133 kbps, and 13 times yields 820 kbps. Forward error correction with redundancy of 5 was selected for the experiment. Storing the input data provides independence of the input and output characteristics. The storage capacity is 24 kilobytes.

The receiving TDB unrandomizes the data, performs the appropriate level of de–interleaving, does a bit–by–bit majority vote, and does the BCH decoding. The data are then passed to the storage buffer where formatting bits are reinserted and then output on the RS–232C line to the data processing device. For more details refer to Harris [1987].

Figure 1 illustrates the experimental configuration.

SOFTWARE CONFIGURATION

The software discussed in this paper consists of four parts: the test driver, the data reduction software, the communications software, and the scenario driver.

The **test driver** is a menu–driven user interface written in C language [Kernighan and Ritchie 1988], the main language of the UNIX operating system [McGilton and Morgan 1983], using the X Window library [Nye 1992], which is based on the X Toolkit Intrinsics, the lower level of programming interface to X [Flanagan 1992], and the Motif toolkit, the standard mechanism on which many of the toolkits written for the X Window System are based [Ferguson 1994]. It coordinates all tasks necessary to execute the experimental design. Prior to the test driver existence, the experimental design for similar tests was executed manually, requiring extra time for setup and the possibility of errors during the initialization phase of a test cell.

Among its tasks, the test driver generates messages for the scenario driver, updates the factor–level combinations, distributes the information to the different nodes, and synchronizes the nodes'

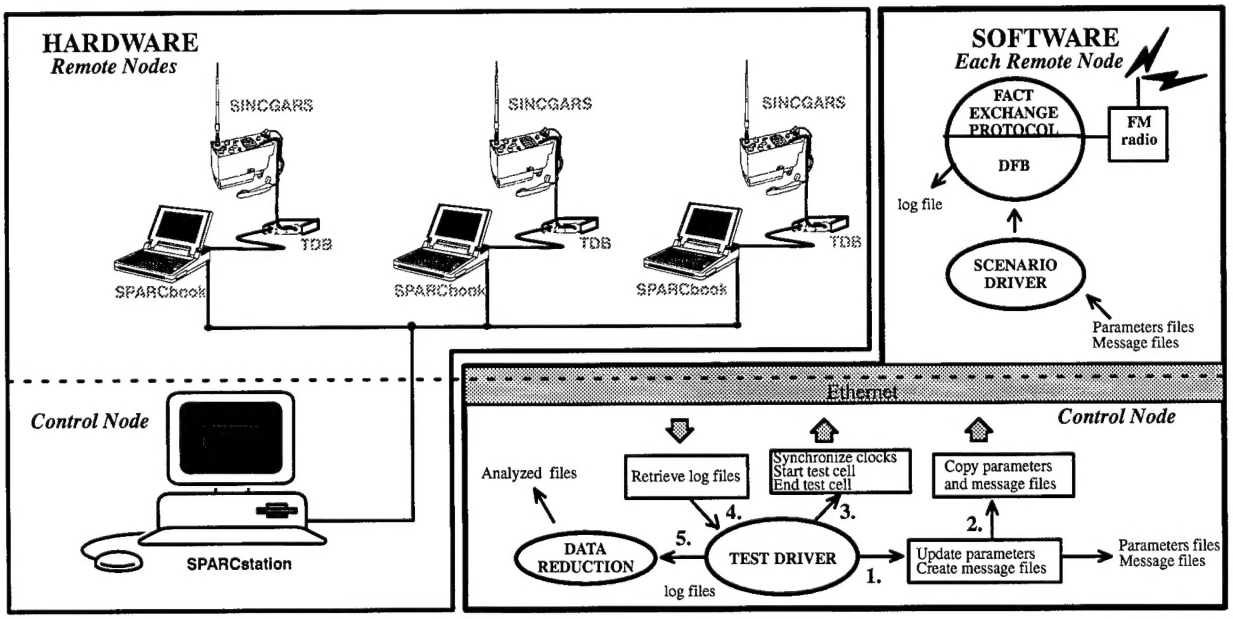


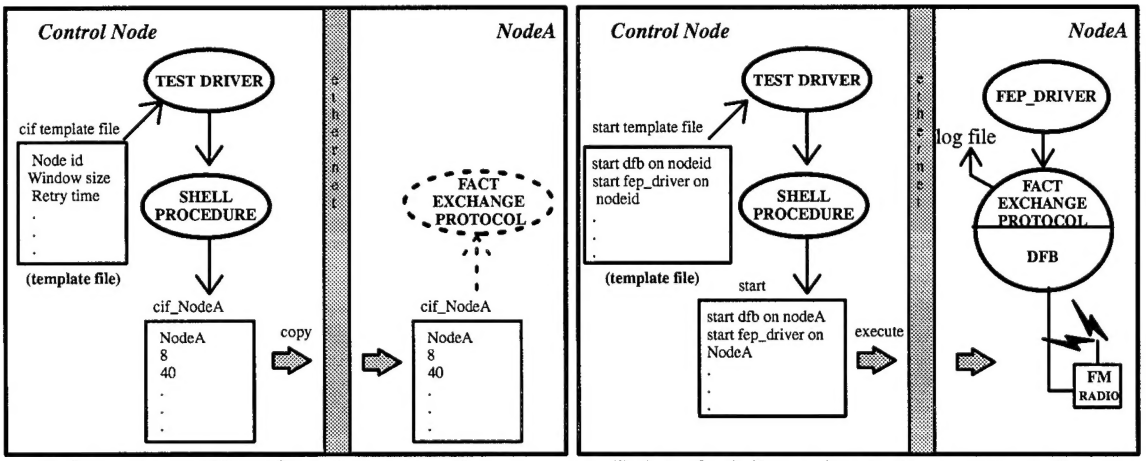
Figure 1. Experimental Configuration

clocks. In addition, it starts and ends each test cell, retrieves all log files from the remote nodes for storage on the control node, and computes measures of performance. To minimize input errors, the test driver runs all experimental combinations without human intervention. The software is capable of executing independent replications of the design matrix automatically, with each replication using different random numbers, starting in the same initial state, and all statistical counters reset to zero.

The test driver reads information contained in text files to initialize values that may vary depending on the experimental design. These text files contain values that need initialization prior to the test cell such as factors and levels of interest, the number of replicates for each test cell, the number of replicates for the center point, the random number seeds to generate the desired message sets or scenarios, the number of retries for each message not delivered during the retry time, node identification string, and the length of each run. Other values that are initialized are the name of the directories into which the software will store the data, the directories where binary files that need execution are located, and values that are used by the data reduction software. The text files used for initialization may be modified either by manually editing the files prior to running the test driver or by menu selection before executing the experimental design.

The communications and scenario driver software on the remote nodes have their own input files, and these also need to be updated prior to each test cell. The control node has a copy of these files (template files), which the test driver updates and copies on the remote nodes. Template files are used whenever part of a file needs to be modified. Examples of this kind of file are the capabilities input file (cif_nodename) loaded by the communications software to initialize the nodes' id, the window size and time to retry (Figure 2a), and the nodename## file from which the scenario driver gets the message information to load messages into communications software.

Tasks executed on the remote node such as synchronizing clocks, starting and ending the execution of a test cell (Figure 2b), as well as on the control node such as copying files to the remote nodes (Figure 2a) and retrieving log files from the remote nodes, are invoked by the test driver through UNIX shell procedures [McGilton and Morgan 1983].



a. Update and copy of input file to the remote node.

b. Update of a shell procedure to start a process on the remote node.

Figure 2. Template File with Shell Procedure Interaction

During the execution of a test cell, each node collects data on a log file local to that node. The log files contain information on the messages and acknowledgments sent and received, as well as information on queues. The **data reduction** software is a set of C programs that reformat log files and compute measures of performance. The test driver executes UNIX shell procedures to invoke the data reduction software. For example, for each log file created by a test cell, the test driver executes a shell procedure *process.s,* which invokes the C program *dr* with a set of arguments to create a file containing the messages transmitted during a particular test cell grouped into 1–minute time intervals. The shell procedure *process.ack* invokes *dr* with different arguments and outputs all the acknowledgments generated during that test cell. The shell procedures that contain node information are updated using template files. The output of the data reduction software is formatted in a fashion suitable for a statistical analysis.

The **communications software** is a C language application composed of a freeform database management system called Distributed FactBase (DFB), which communicates with the other DFBs via the Fact Exchange Protocol (FEP). An important concept implemented in the DFB is the ability to automatically initiate predefined actions (rules) upon receipt of new information. These rules ensure that only significant data (as defined by the commander and staff) are transmitted [Chamberlain 1990]. The FEP is a tactical transport layer protocol that communicates information quickly, concisely, and reliably over noisy nets with minimum bandwidth usage. It is designed to be a connectionless, reliable protocol that utilizes multicast, overhearing, and other techniques to minimize transmissions [Kaste 1990]. A data collection function is provided by the DFB to log information on messages, including acknowledgments, transmitted and received.

The **scenario driver** is a C language application that reads a file of time tagged, preformatted message strings and forwards them to the DFB at the appropriate times.

Figure 3 illustrates the software configuration.

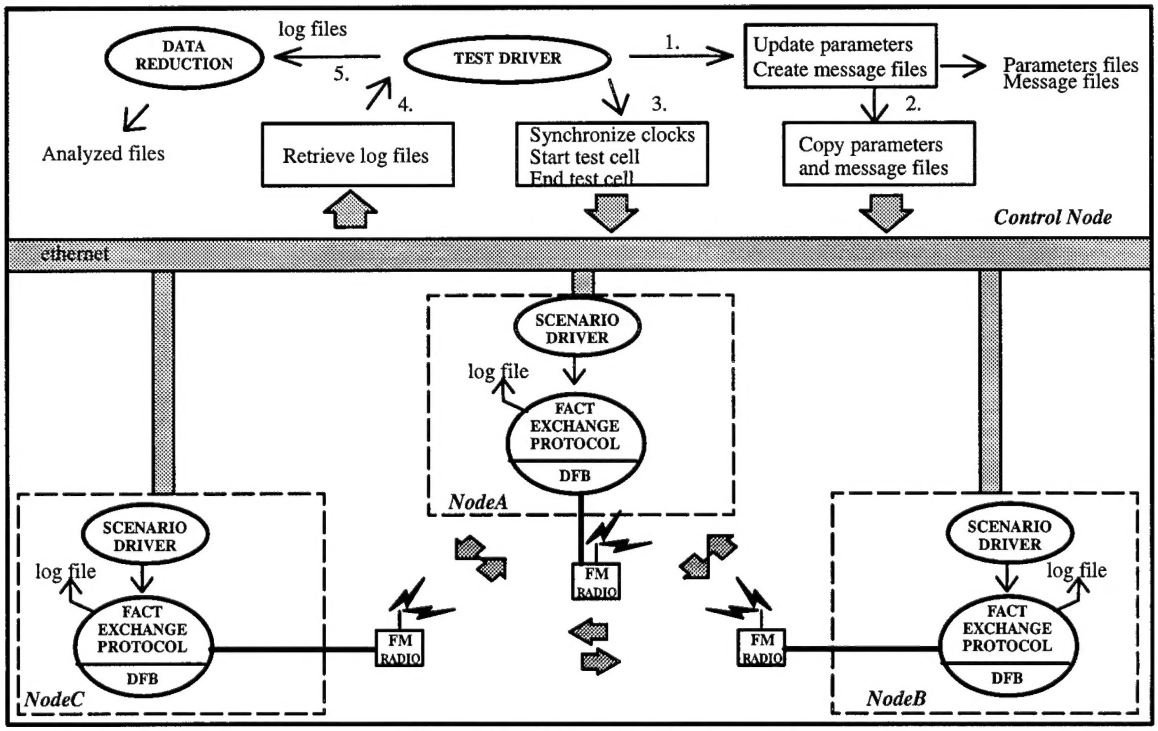


Figure 3. Software Configuration

SUMMARY

The template files are useful in simplifying the programmer's job when the experimental configuration requires modification. This allows fast and easy modifications to experimental configuration since the input is not "hard wired" in the code. For instance, if the number of nodes needs to be increased or decreased, the programmer modifies the input text files containing node information and the updates on the remote software take place during the test driver initialization phase.

Because the test driver is of a general nature, it can be used in a variety of situations to run an experiment in a distributed UNIX environment.

It is anticipated that future experiments can be automated to consider more complex communications protocol modifications. Automating the process reduces the chance of operator error and simplifies the execution of the experimental design.

References

[1] Chamberlain, S. C. "The Information Distribution System: IDS – An Overview." BRL–TR–3114, Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1990.

[2] Ferguson, P. M. Motif Reference Manual. Volume 6B, 1st Edition, Sebastopol, CA: O'Reilly & Associates, Inc., 1994.

[3] Flanagan, D. X Toolkit Intrinsics Reference Manual. Volume 5, 3rd Edition, Sebastopol, CA: O'Reilly & Associates, Inc., 1992.

[4] Harris RF Communications RF–3490 Digital Data Buffer Instruction Manual. Rochester, NY: Harris Corporation, 1987.

[5] Kaste, V. A., A. E. Brodeen and B. D. Broome. "An Experiment to Examine Protocol Performance Over Combat Net Radios." BRL–MR–3978, Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1992.

[6] Kaste, V. A. "The Information Distribution System: The Fact Exchange Protocol, A Tactical Communications Protocol." BRL–MR–3856, Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1990.

[7] Kernighan, B. W., and D. M. Ritchie. The C Programming Language. 2nd Edition, Englewood Cliffs, NJ: Prentice–Hall Inc., 1988.

[8] McGilton, H. and R. Morgan. Introducing the UNIX System. New York, NY: McGraw–Hill Book Company, 1983.

[9] Montgomery, D. C. Design & Analysis of Experiments. 3rd Edition, New York, NY: John Wiley & Sons Inc., 1991.

[10] Nye, A. Xlib Reference Manual. Volume 2, 3rd Edition, Sebastopol, CA: O'Reilly & Associates Inc., 1992.

[11] SPARCbook 3 Series Technical Reference Manual. Austin, TX: Tadpole Technology Inc., 1994.

[12] SPARCstation 20 System Specifications Manual. Mountain View, CA: Sun Microsystems Inc., 1994.